Homework 1

Solution file should be named by Student Number+Name(学号+姓名)

Send solution file to Tutor befor November 2.

Tutor information: Xiongpeng Hu (胡雄鹏), Email: 51215902089@stu.ecnu.edu.cn

Problem 1-1 Asymptotic Notation

For each of the following statements, decide whether it is **always true, never true, or sometimes true** for asymptotically nonnegative functions f and g. If it is **always true or never true**, explain why. If it is **sometimes true**, give one example for which it is true, and one for which it is false.

(1) $f(n) = O(f(n)^2)$
(2) $f(n) + g(n) = \Theta(\max(f(n), g(n)))$
(3) $f(n) + O(f(n)) = \Theta(f(n))$
(4) $f(n) = \Omega(g(n))$ and $f(n) = o(g(n))$
(5) $f(n) \neq O(g(n))$ and $g(n) \neq O(f(n))$

Problem 1-2 Recurrences

Give asymptotic upper and lower bounds for $T(n)$ in each of the following recurrences. Assume that $T(n)$ is constant for $n \leqslant 10$. Make your bounds as tight as possible, and justify your answers.

(1) $T(n) = 2\,T(n/3) + n \lg n$
(2) $T(n) = 3\,T(n/5) + \lg^2 n$
(3) $T(n) = T(n/2) + 2^n$
(4) $T(n) = T(\sqrt{n}) + \Theta(\lg \lg n)$
(5) $T(n) = 10T(n/3) + 17n^{1.2}$
(6) $T(n) = 7T(n/2) + n^3$

Problem 1-3 Sorting an almost sorted list

On his way back from detention, Harry runs into his friend Hermione. He is upset because Professor Snape discovered that his sorting spell failed. Instead of sorting the papers correctly, each paper was within k slots of the proper position. Hermione immediately suggests that insertion sort would have easily fixed the problem. In this problem, we show that Hermione is correct (as usual). As before, $A[1\ldots n]$ in an array of n distinct elements.

(1) First, we define an "inversion." If $i < j$ and $A[i] > A[j]$, then the pair (i,j) is called an inversion of A. What permutation of the array $\{1, 2, \ldots, n\}$ has the most inversions? How many does it have?

(2) Prove that, if every paper is initially within k slots of its proper position, insertion

sort runs in time O(nk). *Hint*: First, prove that INSERTION-SORT(A) runs in time O($n + I$), where I is the number of inversions in A.
(3) Show that sorting a list in which each paper is within k slots of its proper position takes Ω (n lg k) comparisons. *Hint*: Use the decision-tree technique.

Problem 1-4 Edit distance
Many word processors and keyword search engines have a spelling correction feature. If you type in a misspelled word x, the word processor or search engine can suggest a correction y. The correction y should be a word that is close to x. One way to measure the similarity in spelling between two text strings is by "edit distance." The notion of edit distance is useful in other fields as well. For example, biologists use edit distance to characterize the similarity of DNA or protein sequences.

The edit distance $d(x, y)$ of two strings of text, $x[1\,..\,m]$ and $y[1\,..\,n]$, is defined to be the minimum possible cost of a sequence of "transformation operations" (defined below) that transforms string $x[1\,..\,m]$ into string $y[1\,..n]$. To define the effect of the transformation operations, we use an auxiliary string $z[1\,..\,s]$ that holds the intermediate results. At the beginning of the transformation sequence, $s = m$ and $z[1\,..\,s] = x[1\,..\,m]$ (i.e., we start with string $x[1\,..\,m]$). At the end of the transformation sequence, we should have $s = n$ and $z[1\,..\,s] = y[1\,..\,n]$ (i.e., our goal is to transform into string $y[1.\,.n]$). Throughout the transformation, we maintain the current length s of string z, as well as a cursor position i, i.e., an index into string z. The invariant $1 \leqslant i \leqslant s + 1$ holds at all times during the transformation. (Notice that the cursor can move one space beyond the end of the string z in order to allow insertions at the end of the string.)

Each transformation operation may alter the string z, the size s, and the cursor position i. Each transformation operation also has an associated cost. The cost of a sequence of transformation operations is the sum of the costs of the individual operations on the sequence. The goal of the edit-distance problem is to find a sequence of transformation operations of minimum cost that transforms $x[1.\,.\,m]$ into $y[1.\,.n]$. There are five transformation operations:

Operation	**Cost**	**Effect**
left	0	If $i = 1$ then do nothing. Otherwise, set $i \leftarrow i-1$.
right	0	If $i = s + 1$ then do nothing. Otherwise, set $i \leftarrow i+1$.
replace	4	If $i = s+1$ then do nothing. Otherwise, replace the character under the cursor by another character c by setting $z[i] \leftarrow c$, and then incrementing i.
delete	2	If $i = s+1$ then do nothing. Otherwise, delete the character c under the cursor by setting $z[i\,..\,s] \leftarrow z[i+1\,..\,s+1]$ and decrementing s. The cursor position i does not change.
insert	3	Insert the character c into string z by incrementing s, setting $z[i+1\,..\,s] \leftarrow z[i\,..\,s-1]$, setting $z[i] \leftarrow c$, and then incrementing

index *i*.

As an example, one way to transform the source string algorithm to the target string analysis is to use the sequence of operations shown in following table, where the position of the underlined character represents the cursor position *i*. Many other sequences of transformation operations also transform *algorithm* to *analysis*—the solution in table is not unique—and some other solutions cost more while some others cost less.

Operation	z	Cost	Total
Initial string	$\underline{a}lgorithm$	0	0
right	$a\underline{l}gorithm$	0	0
right	$al\underline{g}orithm$	0	0
replace by *y*	$aly\underline{o}rithm$	4	4
replace by *s*	$alys\underline{r}ithm$	4	8
replace by *i*	$alysi\underline{i}thm$	4	12
replace by *s*	$alysis\underline{t}hm$	4	16
delete	$alysis\underline{h}m$	2	18
delete	$alysis\underline{m}$	2	20
delete	$alysis\underline{\ }$	2	22
left	$alysi\underline{s}$	0	22
left	$alys\underline{i}s$	0	22
left	$aly\underline{s}is$	0	22
left	$al\underline{y}sis$	0	22
left	$a\underline{l}ysis$	0	22
insert *n*	$an\underline{l}ysis$	3	25
insert *a*	$ana\underline{l}ysis$	3	28

(1) It is possible to transform *algorithm* to *analysis* without using the "left" operation. Give a sequence of operations in the style of above table that has the same cost as in above table but does not use the "left" operation.
(2)_Describe a dynamic-programming algorithm that computes the edit distance from $x[1 \mathinner{..} m]$ and $y[1 \mathinner{..} n]$. (Do not use a memoized recursive algorithm. Do not use "left" operation. Your algorithm should be a classical, bottom-up, tabular algorithm.) Analyze the running time and space requirements of your algorithm.